CSN08x14

Scripting for Cybersecurity and Networks

Lecture 0a: Python background

About this lecture

- These slides provide some additional background information about Python, its history and relevance, and scripting languages in general.
- They are not directly linked to any of the practicals.
- Read through this in your own time.

A big THANK YOU to **Rich Macfarlane** whose material this module is heavily based on!

Scripting languages

Differences between programming and scripting
What is a scripting language?
Types of scripting language

What is a scripting language?

Python is generally classified as a scripting language. So what does that mean?

- Scripts started out to automate tasks such as command line OS programs, or web page creation
- Used to write **'scripts'** - 'fast and light'
- Typically interpreted and not compiled
- Smaller programs – hundreds of lines max
- Dynamic Typing – allows rapid development
- High level language – higher than C#, Java

Programming vs Scripting

Programming languages	Scripting languages
• Generally strict syntax rules • Usually compiled • Often produce applications that run independently	• Often less strict about syntax • Usually interpreted • Often for applications that interface with other applications (e.g. with a web server, web browser) • Tend to do more with fewer lines of code • May be easier to learn • Good for rapid prototyping

To read a fully written explanation: Differences Between Programming vs Scripting
https://www.educba.com/programming-vs-scripting/

Programming vs scripting

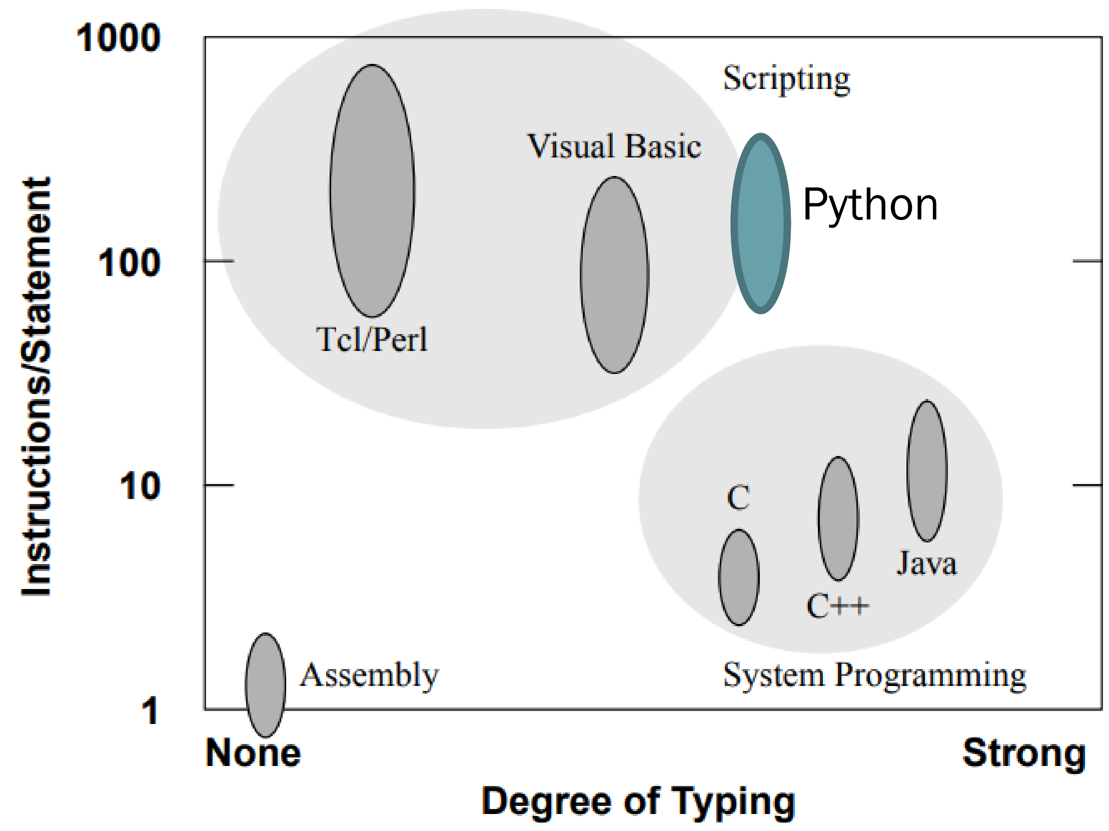


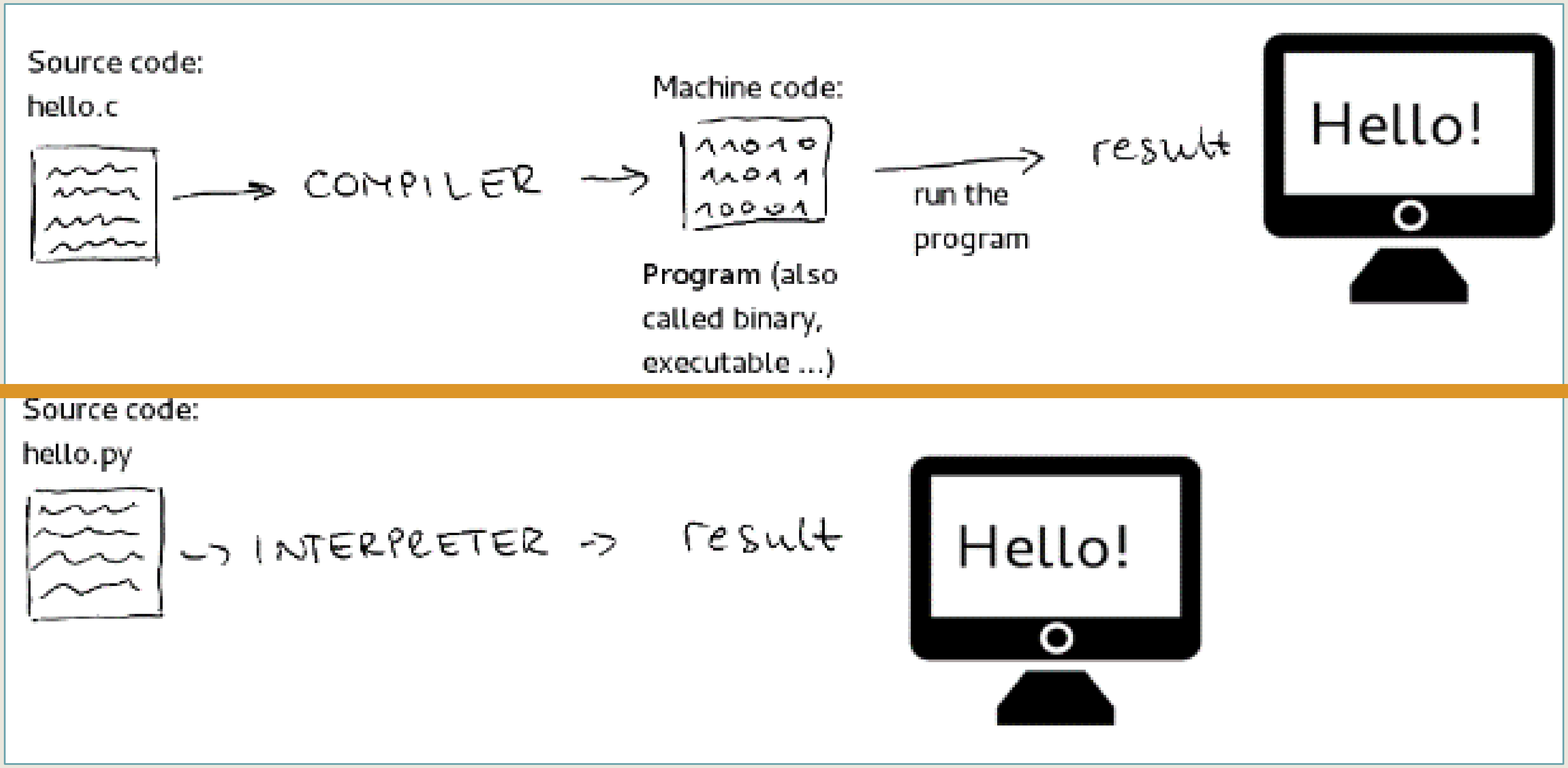
Figure 1. A comparison of various programming languages based on their level (higher level languages execute more machine instructions for each language statement) and their degree of typing. System programming languages like C tend to be strongly typed and medium level (5-10 instructions/statement). Scripting languages like Tcl tend to be weakly typed and very high level (100-1000 instructions/statement).

From: John K. Ousterhout (1997). Scripting: Higher Level Programming for the 21st Century https://pdfs.semanticscholar.org/adf1/67bfead0e0717ba2de1a8067cf403a2a05ba.pdf

Scripting Language Types

- OS Job control – JCL, Unix Shell Scripting, Bash
- Unix/Linux shell scripting – glue programs together:
  –
  ```
  ps –aux | grep “snort” | more
  ```
- Text Processing – AWK, sed - Perl (started as)
- Web browser control – JavaScript
  - *Scripts embedded in HTML – provide dynamic content*
- Web pages - PHP
- Developed into application programming languages – Perl, Python, Ruby
  - *Now Modular and Extensible, OO*

Compiled vs interpreted languages

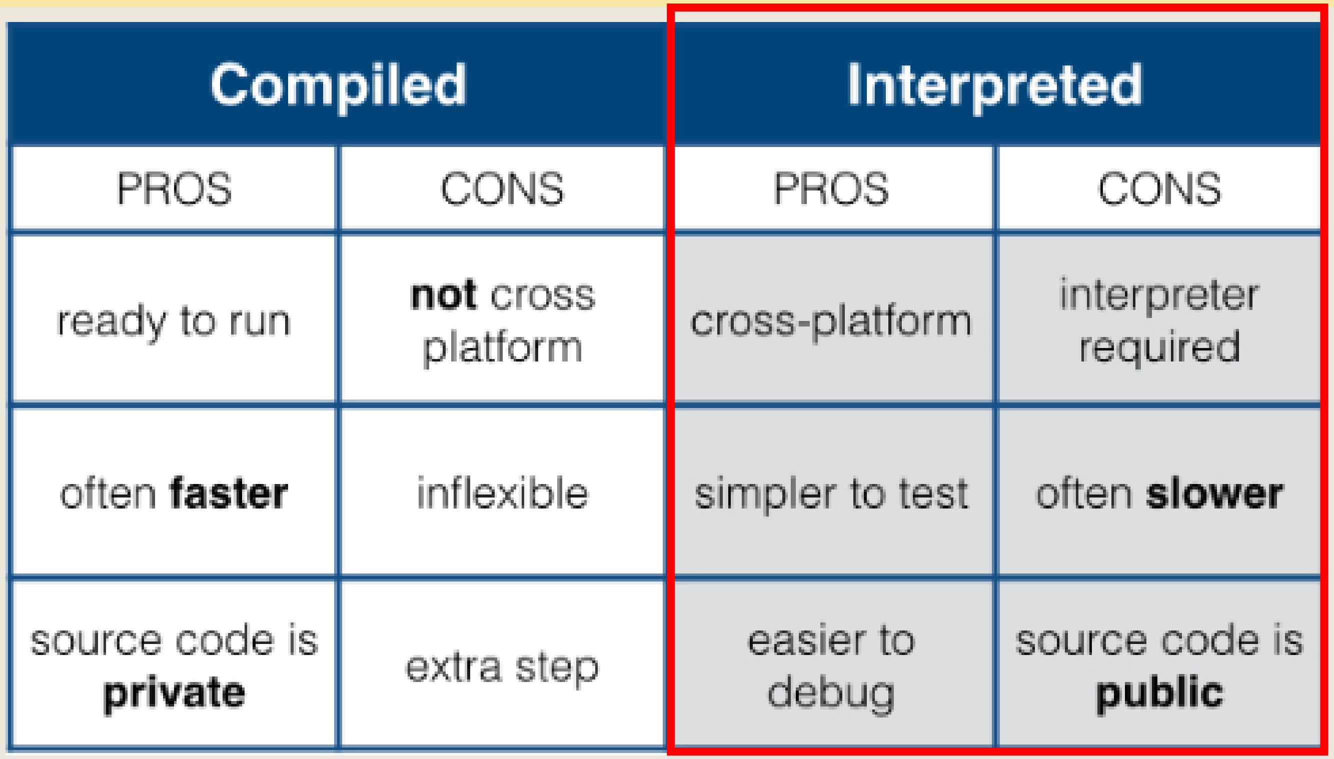


- Compile: translate entire code into machine code (executable).
- Platform dependent

- Interpret: translates code into machine code line by line while script is running
- Platform independent
- Slower at runtime

Image credit http://ikajic.github.io/starting-with-python/#/7

Compiled vs interpreted languages



Compiled		Interpreted	
PROS	CONS	PROS	CONS
ready to run	**not** cross platform	cross-platform	interpreter required
often **faster**	inflexible	simpler to test	often **slower**
source code is **private**	extra step	easier to debug	source code is **public**

Table from https://learntocodewith.me/programming/source-code/

Python is usually called an Interpreted language - But could be called a hybrid language (like Java)

- Automatically pre-processes the source code
- stored as .pyc file (If pyc exists and is current, used automatically)
- Platform independent byte code (not machine code).
- Easy to reverse engineer (source code remains accessible)
- .pyc improves load time but not run time

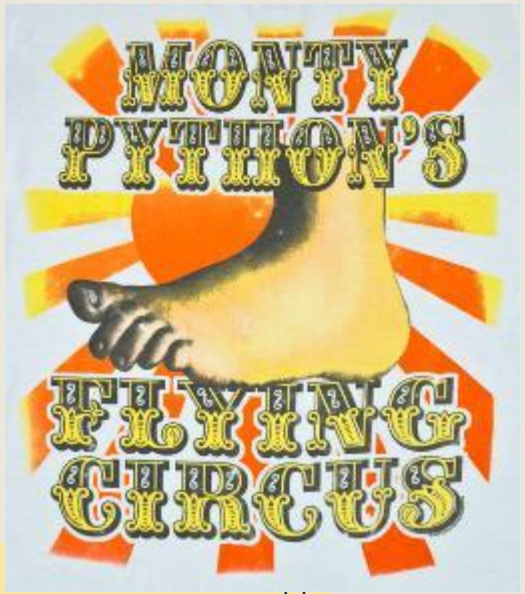
…but you can use e.g. PyInstaller to create a stand-alone exe for distribution if you want to

Python overview

Python History
Why learn Python?
What is Python used for?
Who uses Python?

What is Python?

- **Powerful/Light High Level Language**
- Scripting/general purpose programming language
- Between traditional scripting languages (TCL, Perl, bash) and high level application development languages (C++, C#, Java)
- Similar to Perl, Ruby

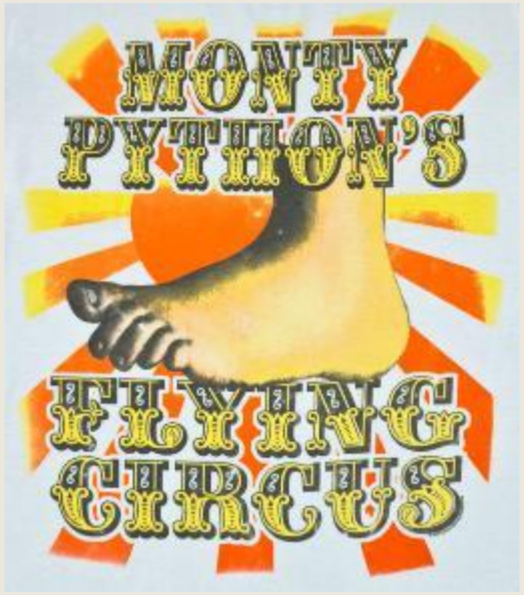


Python History

- Python created in **1989** by **Guido van Rossum**
  (previously at Google in App Engine Team and various other places.
  Currently works for Dropbox – client/server written in Python)
- Python is named after **Monty Python's Flying Circus**

Guido van Rossum

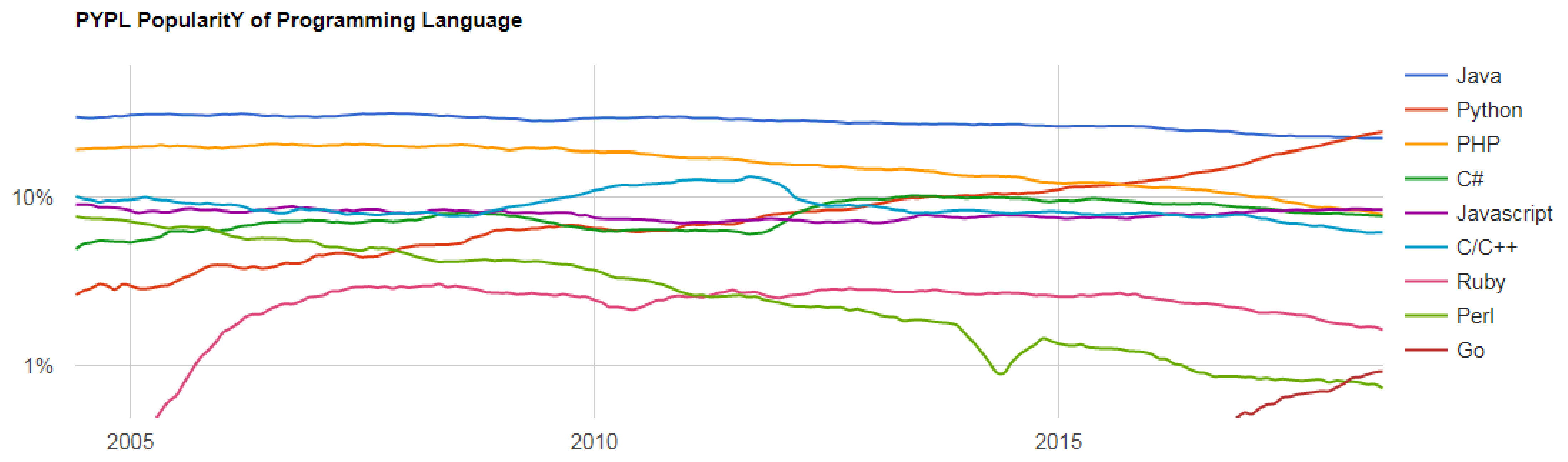
...and his number plate



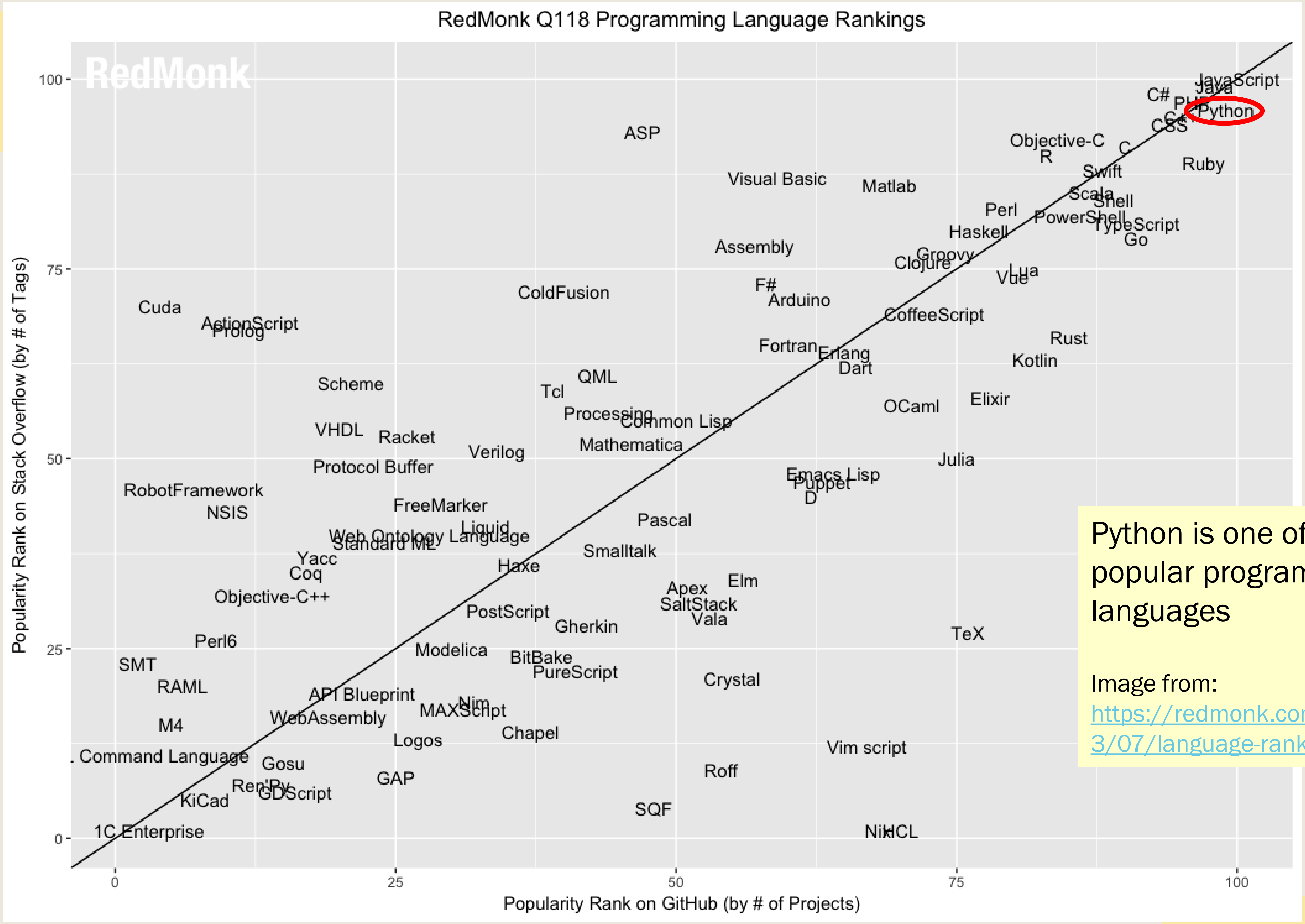
Python History

- Not a new language – but keeps gaining popularity

Worldwide, Python is the most popular language, Python grew the most in the last 5 years (14.3%) and PHP lost the most (-6.5%)



See http://pypl.github.io/PYPL.html



Python is one of the most popular programming languages

Image from: https://redmonk.com/sogrady/2018/03/07/language-rankings-1-18/

Python Features

- Interpreted Language
  - *No compile phase/quick turnaround*
  - *Prototype code in Interpreter before adding to scripts*
- Dynamically Typed
  - *Typing at runtime - as apposed to static compile type typing:*
- Strongly Typed
  - *Dynamic type checking – reports misuse*
- Automated Memory Management
  - *Garbage collection*
- Simplicity of Scripting – procedural
- Object Oriented - optional
- Language, with SE tools and libraries of high level compiled languages

Python Platforms

- **Same code will run on:**
  - *Linux – Python installed on most*
  - *Mac – pre installed*
  - *Windows*
  - *Mobile – Android/iOS*
  - *Games Consoles*
  - *Embedded Systems*

- **Implementations**:
  - *Cpython – reference implementation*
  - *IronPython – for scripting .NET and Mono*
  - *Jpython – coded for Java Runtime*

Python Applications

- Operating-Systems programming
  - *OS Shell tools, scripting*
  - *Cross platform interfaces to OS*
- Cyber Security and Forensics
  - *Security –pen testing, exploit creation, automation of tools, text parsing, log file analysis*
  - *Forensics – tools, scripting - digital media analysis*
- Network scripting
  - *Packet Sniffing, Traffic analysis – Scapy, pyNIDS*
- Internet scripting
  - *Client/server networking, XML, Fetch web pages, parse HTML*
  - *Web application development – django*
- GUI programming - tkinter
- Database programming – cross db libraries
- Data Mining/Machine Learning, Robotics, Gaming
- Numeric/Scientific Programming – NumPy, SciPy

Python Commercial User Base

- Intel, Cisco, HP
- Google, Yahoo
- YouTube
- Industrial Light & Magic
- BitTorrent
- NASA
- Reddit
- Raspberry Pi
- NSA

Python principles

Python Principles ‘easter egg’:

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Version 3.x (3.7 released June 2018)

- Most libraries now support 3.x
- Most textbooks and tutorial sites are for 3.x
- Eliminates old quirks that confuse learners
- Full Unicode support
  - *Clear distinction between strings and binary*
- New for 3.6:
  - *f-string formatting*
  - *secrets module*
  - *Underscores allowed in variable names*
  - *Type hints for named variables*

Version 2.7 - legacy

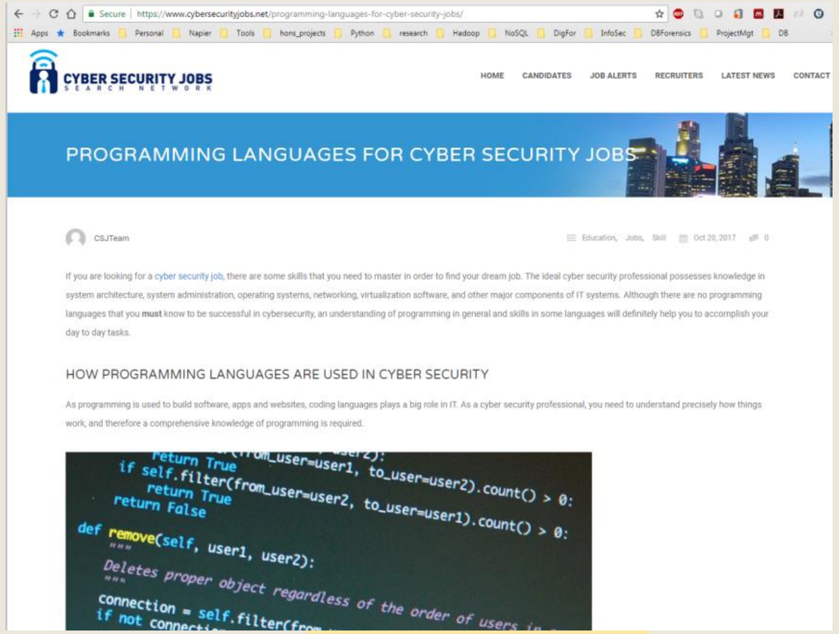
- Used by many existing applications
- Still default in some OS
- Support will be discontinued on 1/1/2020 http://pythonclock.org/

We will use the current version of **Python 3.x** in this module
https://wiki.python.org/moin/Python2orPython3
http://python-notes.curiousefficiency.org/en/latest/python3/questions_and_answers.html#why-is-python-3-considered-a-better-language-to-teach-beginning-programmers

Why Python?

Why are we learning Python?

- Cybersecurity pros need programming skills
  - *Automate repetitive tasks*
  - *Analyse software for vulnerabilities*
  - *Identify malicious software*
  - *Script your own tools – e.g. plug ins for forensic software*
- Python is possibly <u>the</u> most popular language for cybersecurity
  - *Can be used for wide range of applications*
  - *Open source*
  - *Easy to learn*
  - *Rich set of libraries/tools available*
  - *Platform independent*


https://www.cybersecurityjobs.net/programming-languages-for-cyber-security-jobs/
CYBER SECURITY JOBS SEARCH NETWORK
HOME CANDIDATES JOB ALERTS RECRUITERS LATEST NEWS CONTACT

PROGRAMMING LANGUAGES FOR CYBER SECURITY JOBS

CSJTeam Education, Jobs, Skill Oct 20, 2017 0

If you are looking for a cyber security job, there are some skills that you need to master in order to find your dream job. The ideal cyber security professional possesses knowledge in system architecture, system administration, operating systems, networking, virtualization software, and other major components of IT systems. Although there are no programming languages that you **must** know to be successful in cybersecurity, an understanding of programming in general and skills in some languages will definitely help you to accomplish your day to day tasks.

HOW PROGRAMMING LANGUAGES ARE USED IN CYBER SECURITY

As programming is used to build software, apps and websites, coding languages plays a big role in IT. As a cyber security professional, you need to understand precisely how things work, and therefore a comprehensive knowledge of programming is required.

```
        return True
    if self.filter(from_user=user2, to_user=user1).count() > 0:
        return True
    return False

def remove(self, user1, user2):
    """
    Deletes proper object regardless of the order of users in
    """
    connection = self.filter(from
    if not connection
```

https://www.cybersecurityjobs.net/programming-languages-for-cyber-security-jobs/

Why are we learning Python?

- Designed to help produce Good Quality Code
  - *Readable/consistent/minimal code compared to other scripting languages – more readable than Perl*
  - *Easy to learn and use – easier than C, C++ ,C#, Java*
- Productivity
  - *Rapid Prototyping/Proof of concept - hacking*
  - *Less code to type/no compile stage*
- Portability
  - *Multi platform without any changes*
  - *Windows/Linux/Mac/Mobile*

Why not Perl?

- 'Conflict' between Language Users
- Can do everything in Python that you can do in Perl
- Perl code shorter, quicker to code, but typically less readable, and harder to reuse
- Perl typically more ways to do same thing
- Python typically has only one(ish) way to do things

Python vs Perl (according to Yoda)

```
YODA: Code!  Yes.  A programmer's strength flows from code
      maintainability.  But beware of Perl.  Terse syntax... more
      than one way to do it...  default variables.  The dark side
      of code maintainability are they.  Easily they flow, quick
      to join you when code you write.  If once you start down the
      dark path, forever will it dominate your destiny, consume
      you it will.

LUKE: Is Perl better than Python?

YODA: No... no... no.  Quicker, easier, more seductive.

LUKE: But how will I know why Python is better than Perl?

YODA: You will know.  When your code you try to read six months
      from now.
```

Attributed to Larry Hastings, Quoted in Mark Lutz, Programming Python, 3rd ed (2006), p.1524.